DOCKET NO.: 3829-020-27

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Marco SASSELLI, et al. ART UNIT:
SERIAL NO: New Application EXAMINER:
FILED: Herewith
FOR: METHOD FOR SECURING SOFTWARE UPDATES

REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
PO BOX 1450
ALEXANDRIA, VA 22313-1450

SIR:

- ☐ Full benefit of the filing date of U.S. Application Serial Number [US App No], filed [US App Dt], is claimed pursuant to the provisions of **35 U.S.C. §120**.
- ☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §119(e)**.
- ■ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

COUNTRY	APPLICATION NUMBER	MONTH/DAY/YEAR
Switzerland	2043/02	December 3, 2003

Certified copies of the corresponding Convention Application(s)

- ■ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee
- ☐ were filed in prior application Serial No. filed
- ☐ were submitted to the International Bureau in PCT Application Number
- ☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and
  (B) Application Serial No.(s)
  - ☐ are submitted herewith
  - ☐ will be submitted prior to payment of the Final Fee

Respectfully submitted,

PIPER RUDNICK LLP

1200 Nineteenth Street, N.W.
Washington, D.C. 20036-2412
Telephone No. (202) 861-3900
Facsimile No. (202) 223-2085

Steven B. Kelber
Registration No.: 30,073

James M. Heintz
Registration No.: 41,828

SCHWEIZERISCHE EIDGENOSSENSCHAFT
CONFÉDÉRATION SUISSE
CONFEDERAZIONE SVIZZERA

Bescheinigung

Die beiliegenden Akten stimmen mit den ursprünglichen technischen Unterlagen des auf der nächsten Seite bezeichneten Patentgesuches für die Schweiz und Liechtenstein überein. Die Schweiz und das Fürstentum Liechtenstein bilden ein einheitliches Schutzgebiet. Der Schutz kann deshalb nur für beide Länder gemeinsam beantragt werden.

Attestation

Les documents ci-joints sont conformes aux pièces techniques originales de la demande de brevet pour la Suisse et le Liechtenstein spécifiée à la page suivante. La Suisse et la Principauté de Liechtenstein constituent un territoire unitaire de protection. La protection ne peut donc être revendiquée que pour l'ensemble des deux Etats.

Attestazione

I documenti allegati sono conformi agli atti tecnici originali della domanda di brevetto per la Svizzera e il Liechtenstein specificata nella pagina seguente. La Svizzera e il Principato di Liechtenstein formano un unico territorio di protezione. La protezione può dunque essere rivendicata solamente per l'insieme dei due Stati.

Bern, 14. NOV. 2003

Eidgenössisches Institut für Geistiges Eigentum
Institut Fédéral de la Propriété Intellectuelle
Istituto Federale della Proprietà Intellettuale

Patentverfahren
Administration des brevets
Amministrazione dei brevetti

H. Jenni

Heinz Jenni

Office of Intellectual Property Law

Demande de brevet no 2002 2043/02

CERTIFICAT DE DEPOT (art. 46 al. 5 OBI)

L'Institut Fédéral de la Propriété Intellectuelle accuse réception de la demande de brevet Suisse dont le détail figure ci-dessous.

Titre:
Méthode de sécurisation des mises à jour de logiciels.

Requérant:
Nagravision S.A.
22, Route de Genève
1033 Cheseaux-sur-Lausanne

Mandataire:
Leman Consulting S.A.
62 rte de Clementy
1260 Nyon

Date du dépôt: 03.12.2002

Classement provisoire: G06K, G09C

MÉTHODE DE SÉCURISATION DES MISES À JOUR DE LOGICIELS

La présente invention concerne une méthode de sécurisation des mises à jour de logiciels d'exploitation assurant le fonctionnement de systèmes les plus divers. Plus particulièrement, la méthode de l'invention utilise un mécanisme de signature numérique avec une clé privée d'un algorithme d'encryptage asymétrique.

Un système est défini ici comme un appareil ou un ensemble d'appareils dont le fonctionnement dépend d'un ou plusieurs logiciels stockés dans une mémoire non volatile ou un disque dur. Lorsque les fonctionnalités du système doivent être améliorées ou complétées afin de s'adapter aux exigences croissantes de l'utilisateur, il est souvent nécessaire de mettre à jour uniquement le logiciel concerné sans pour autant changer le matériel constituant le système.

La mise à jour d'un logiciel donné s'effectue en général par remplacement de fichiers d'un logiciel déjà installé ou par ajout de nouveaux fichiers venant compléter ceux qui sont stockés. L'ensemble constitue alors une nouvelle version du logiciel préalablement installé dans le système qui bénéficie ainsi des améliorations souhaitées.

De nombreux appareils tels que des ordinateurs et leurs périphériques, des automates de vente, des téléphones fixes et portables, des décodeurs de télévision à péage, etc. sont pilotés par des logiciels adaptés à leur configuration et aux fonctions de composants spécifiques.

Par exemple, dans un décodeur de télévision à péage (Set Top Box), un logiciel d'exploitation gère des périphériques tels qu'un disque dur, un lecteur de cartes à puce, des interfaces de réception de données et des mémoires. Afin d'introduire des changements soit au niveau de la configuration, soit au niveau des fonctionnalités, il est parfois nécessaire de remplacer le logiciel existant ou d'apporter des améliorations à celui déjà installé dans le décodeur. Ce type d'évolution s'effectue au moyen de portions de logiciels que l'on appelle mises à jour ou patchs fournis par le centre de gestion de l'opérateur auquel un certain nombre d'utilisateurs sont abonnés. Ces mises à jour sont fournies régulièrement par le centre de gestion et téléchargées dans les décodeurs de chaque abonné possédant les droits nécessaires.

Les utilisateurs d'un décodeur possèdent en général un contrat payant chez un opérateur qui leur garantit un service de maintenance régulier du logiciel installé dans le décodeur. Afin de limiter les abus par copies non autorisées et par introduction de composants logiciels étrangers, il devient indispensable de sécuriser les mises à jour du logiciel du décodeur. Une méthode connue consiste à utiliser une empreinte codée avec une clé d'un algorithme d'encryptage asymétrique du type RSA. Le logiciel de mise à jour est fourni en ligne avec une empreinte obtenue par une fonction de hachage à sens unique (Hash). Cette empreinte constitue une image unique représentant l'ensemble de la mise à jour et il est admis qu'il n'existe pas deux empreintes identiques sur deux même ensembles de données différents. Cette empreinte est encryptée à l'aide d'une clé privée de l'opérateur associée à un ensemble d'abonnés, ce qui constitue une signature propre à cet ensemble. Le logiciel accompagné de cette signature est chargé dans une mémoire vive RAM (Random Access Memory) du décodeur. Un programme faisant partie du logiciel du décodeur calcule avec la fonction de hachage (Hash) une empreinte du logiciel stocké dans la mémoire RAM. La signature reçue est décryptée avec une clé publique contenue dans le décodeur puis comparée avec l'empreinte du logiciel calculée auparavant. Si la signature décryptée correspond à l'empreinte résultante du hachage, la signature accompagnant la mise à jour stockée dans la mémoire RAM est considérée comme valide. Le logiciel de mise à jour sera installé dans la mémoire non volatile du décodeur (mémoire Flash).

La sécurisation est ainsi réalisée par la vérification de la signature avec une clé publique dans le décodeur correspondant à la clé privée de l'opérateur.

La clé publique résidant dans le décodeur doit être fixe ainsi que le programme qui permet la vérification de la signature. L'authenticité de la signature est assurée du fait que la clé privée dépend de la clé publique du décodeur. La signature ne peut pas être reproduite car la clé privée n'est connue que d'un opérateur déterminé. De plus, une même signature est inutilisable pour plusieurs mises à jour différentes car elle est une fonction d'une mise à jour bien définie. Un logiciel de mise à jour signé et dont le contenu est modifié dans la mémoire RAM donnera une autre empreinte qui ne peut donc pas être validée par la signature décryptée par la clé publique. En effet,

l'empreinte résultante après hachage de la mise à jour au niveau du décodeur et différente de celle obtenue après décryptage de la signature.

Cependant, cette méthode de sécurisation comporte un point faible qui est la clé privée de l'opérateur elle-même. En effet, lorsque celle-ci est découverte par un tiers, il peut signer un logiciel quelconque et apporter des modifications abusives au système.

Cette découverte peut se faire sous forme d'itération sur la clé publique que ce tiers aura extraite au décodeur, jusqu'à ce qu'il découvre la bonne paire de clés. La parade consistant à modifier le comportement le logiciel du décodeur pour qu'il refuse les signatures générées avec la clé découverte est insuffisante car le tiers peut contourner ces modifications avec des programmes adéquats.

Le but de la présente invention est de réduire considérablement l'impact de la découverte d'une clé privée par une analyse systématique du fonctionnement du logiciel du décodeur ou d'accroître notablement le temps et les moyens nécessaires au processus utilisé pour sa détermination.

Le but est atteint par une méthode de sécurisation de mise à jour de données d'une pluralité d'appareils, chaque appareil recevant les mises à jour d'un centre de gestion, ces mises à jour comprenant des données appelées patch accompagnées d'un bloc de contrôle encrypté par une clé privée asymétrique prise dans une liste de clés contenue dans le centre de gestion, caractérisée par les étapes suivantes:

- sélection par l'appareil d'une clé courante dans une liste de clés publiques,
- réception du patch de mise à jour et stockage en mémoire,
- réception du bloc de contrôle encrypté,
- décryption dudit bloc par la clé publique courante,
- vérification que le bloc de contrôle décrypté corresponde audit patch,
- installation du patch reçu,
- désactivation de la clé courante et sélection de la clé suivante dans la liste.

Les données d'une mise à jour sont transmises par le centre de gestion sous forme d'un patch et d'un bloc de contrôle comprenant une signature constituée par

l'empreinte du patch encryptée avec une clé privée du centre de gestion. Le décodeur stocke ces données dans la mémoire vive RAM en vue de leur traitement. Une clé publique, associée à cette clé privée appelée clé courante, est sélectionnée à partir d'une liste stockée dans une première mémoire non volatile afin de décrypter la signature du patch. En cas de succès de la décryption et de la vérification, une commande est exécutée aboutissant à l'installation du patch dans une seconde mémoire non volatile (Flash) du décodeur. La clé courante ainsi utilisée est désactivée dans la liste ce qui rend la clé suivante disponible pour la prochaine mise à jour.

Lorsque la vérification de la signature et la décryption d'une mise à jour du décodeur est effectuée avec une clé publique de la liste, celle-ci est biffée et devient inutilisable pour de prochaines mises à jour. Donc, à chaque mise à jour, une nouvelle clé est utilisée puis éliminée de la liste. Les clés publiques de la liste doivent être non modifiables tout comme le programme servant à la vérification des signatures. La liste n'est modifiable (élimination des clés utilisées) que par le programme de vérification.

La méthode décrite ci-dessus permet de réduire considérablement les possibilités de modifications du décodeur par un tiers ayant découvert une clé privée. Puisqu'une clé n'est utilisable qu'une seule fois, le tiers n'effectuera qu'une seule modification. Par conséquent une modification du comportement du décodeur pour le protéger des effets du piratage devient plus efficace car le tiers, ne disposant plus de clé privée valable, se trouve ainsi devant un appareil inaccessible.

Du fait que les clés privées utilisables sont plus nombreuses, un tiers doit donc attaquer systématiquement toutes les clés pour qu'il ne soit pas bloqué à une mise à jour qui correspond à la clé qu'il a découverte. Il faut savoir que l'évolution du logiciel des décodeurs est rapide et que l'on considère que si une des clés est découverte, les mises à jour suivantes vont refermer les failles de sécurité que le tiers aurait pu introduire. Si ce tiers est capable de bloquer toute mise à jour postérieure, les fonctionnalités du décodeur seront rapidement obsolètes et donc sans grand préjudice pour l'opérateur.

L'utilisation de clés asymétriques est importante dans ce contexte car l'extraction des clés publiques dans un décodeur ne permet pas de fabriquer une mise à jour acceptable puisque cette mise à jour doit être signée par la clé privée de l'opérateur. Il est commun de placer les clés privées dans la partie sécurisée (l'opérateur) et les clés publiques dans la partie dans le domaine public (le décodeur). Néanmoins, il est possible d'inverser ces clés sans nuire au fonctionnement de la présente invention.

Un premier mode de réalisation de l'invention propose l'utilisation de clés publiques prises dans un ordre prédéterminé à partir de la liste. Ainsi, chaque clé de la liste est prise dès que la clé précédente est utilisée.

L'invention sera mieux comprise grâce à la description détaillée qui va suivre et qui se réfère aux figures annexées servant d'exemple nullement limitatif, à savoir:

- La figure 1 représente le déroulement d'une étape de mise à jour d'un décodeur d'une version N à une version N+1.

- La figure 2 montre une mise à jour d'une version N vers une version R

Dans l'exemple illustré par la figure 1, un décodeur de version initiale est mis à jour vers la version 1 avec un patch P1. Ce patch P1 est transmis avec sa signature $(H(P1))_{PK1}$ par le centre de gestion de l'opérateur vers le décodeur. L'étape de mise à jour débute par le chargement du patch P1 dans la mémoire RAM du décodeur.

La signature $(H(P1))_{PK1}$ est obtenue par encryption de l'empreinte H(P1) du patch P1 avec la clé privée PK1 de l'opérateur, opération effectuée dans le centre de gestion. Cette empreinte est calculée par l'opérateur à partir du patch P1 avec une fonction de hachage H à sens unique de type Hash.

Le logiciel du décodeur se charge ensuite de décrypter la signature $(H(P1))_{PK1}$ reçue avec une clé publique K1 afin d'obtenir l'empreinte du patch $H(P1)_1$. Parallèlement, ce même logiciel calcule l'empreinte $H(P1)_2$ du patch P1 stockée dans la mémoire RAM. La première empreinte issue de la décryption de la signature $H(P1)_1$ et la seconde $H(P1)_2$ résultante du calcul par la fonction de hachage H sont comparées. Si les deux valeurs concordent, le patch P1 est installé dans la mémoire non volatile Flash FH du décodeur réalisant ainsi la mise à jour du logiciel du décodeur. La clé publique K1 utilisée pour décrypter la signature est biffée de la liste.

Une seconde mise à jour de la version 1 vers la version 2 transmise par le centre de gestion sous forme d'un nouveau patch P2 accompagné de sa signature $(H(P2))_{PK2}$ passe par le même procédé de téléchargement et de vérification. Une nouvelle clé publique K2 prise dans la liste sera alors utilisée du côté du décodeur. Toutes les mises à jour suivantes transmises sont vérifiées de la même manière en utilisant chaque fois une nouvelle clé publique prise dans la liste. Les clés des mises à jour précédentes sont neutralisées soit par effacement, soit par un marquage adéquat.

En appliquant ce procédé, une mise à jour d'un logiciel de la version 1 vers une version N s'effectue en N-1 étapes. Le centre de gestion transmettra N-1 patchs avec N-1 signatures correspondantes encryptées chacune par une clé privée propre à chaque version. L'installation des différents patchs entraîne donc la neutralisation de N-1 clés publiques de la liste.

La liste des clés publiques peut être stockée par exemple dans une mémoire non volatile du type EEPROM (Electrically Erasable Programmable Read-Only Memory). Après chaque utilisation d'une clé lors d'une mise à jour, celle-ci est effacée définitivement de l'EEPROM autorisant l'accès à la clé suivante pour la prochaine mise à jour.

Selon un autre mode de réalisation, la liste des clés publiques n'est pas altérée par un effacement ou un marquage d'une clé. Après chaque installation d'une version de logiciel dans la mémoire non volatile Flash, un compteur est incrémenté ou un pointeur se déplace pour indiquer le rang de la clé à sélectionner dans la liste lors de la prochaine mise à jour. Ainsi lors de chaque mise à jour, seule la clé qui servira à décrypter la signature du patch est désignée, les clés précédentes ne pouvant plus être sélectionnées car le compteur ou le pointeur ne peut progresser que dans un seul sens, celui des rangs croissants.

Selon une variante, le patch peut être encrypté par la clé privée de l'opérateur. Une étape supplémentaire de décryption s'ajoute donc au procédé décrit ci-dessus. Le patch P reçu et chargé dans la mémoire RAM peut être décrypté avec la clé publique avant le calcul par la fonction de hachage H de l'empreinte servant à la vérification de la signature. Le calcul de l'empreinte peut également être effectué sur les patchs sous sa forme encryptée également.

L'installation de mises à jour par des tiers est rendue plus difficile car chaque changement de version exige la connaissance de la clé courante. Cette dernière change à chaque mise à jour ce qui oblige le tiers à connaître toutes les clés pour suivre les différentes mise à jour.

Le procédé décrit précédemment peut poser un problème lorsque le décodeur est resté hors service le temps où plusieurs mises à jour devaient être effectuées. Le passage d'une ancienne version d'un logiciel à une nouvelle dont le numéro n'est pas consécutif à celui de la précédente version s'effectue séquentiellement en plusieurs étapes successives. Ces dernières utilisant des clés publiques différentes prises dans la liste l'une après l'autre et dans l'ordre. Il est rappelé que le patch en lui-même ne contient pas d'ordre permettant de sélectionner une clé différente que la clé courante. Si tel était le cas, un tiers pourrait utiliser cette commande pour forcer l'utilisation d'une clé qu'il lui serait connue.

La figure 2 illustre le cas d'un passage d'un logiciel d'une version N à une version R où la différence R-N entre la nouvelle version et la précédente excède l'unité. L'exemple qui suit se rapporte à un cas où N=2 et R=5.

Un décodeur dont le logiciel est de version 2 ne peut pas décrypter directement la signature $(H(P))_{PK5}$ de la nouvelle version 5 car la clé disponible dans la liste des clés publiques est celle de la version immédiatement supérieure, à savoir la clé K3. Pour l'installation de la nouvelle version 5, il doit pouvoir accéder à la clé correspondante à cette version, c'est à dire la clé K5.

La solution consiste à transmettre un flux de données contenant le patch P pour la mise à jour du logiciel du décodeur à la version 5 signé avec la clé PK5 auquel s'ajoute une pluralité de messages M1, M2, M3, M4 encryptés chacun avec une clé privée PK1, PK2, PK3, PK4 tirée de la liste de clés. La mémoire vive RAM stocke ces messages ainsi que le patch P avec sa signature $(H(P))_{PK5}$. La version du décodeur étant 2, la clé de la mise à jour de la version 1 à 2 est déjà désactivée par la première mise à jour. Le message M1 est alors ignoré car la clé K1 servant à le décrypter n'est plus disponible.

Les messages suivants M2, M3 et M4 servent à désactiver l'une après l'autre les clés publiques K2, K3, et K4 correspondantes à chaque version intermédiaire depuis la

version 2 jusqu'à la version 4 précédant la version 5. Ainsi pour installer la version 5 dans la mémoire non volatile Flash, chaque clé publique K2, K3, et K4 de la liste est utilisée puis neutralisée ou biffée. Lors de la décryption du message par la bonne clé, le contenu de ce message est reconnu et provoque l'opération de neutralisation de la clé courante. Si le message n'est pas reconnu, ceci signifie que la clé d'encryption de ce message n'est pas la clé courante.

Après la décryption successive et réussies des messages M2, M3, M4, la clé K5 nécessaire à la décryption de la signature du patch $(H(P))_{PK5}$ (et du patch P) devient la clé courante. Cette dernière sera aussi biffée de la liste après l'installation du patch et la clé K6 sera présente en tête de liste pour la prochaine mise à jour de la version 5 vers la version 6.

Un tel flux peut donc mettre à jour tout un parc de décodeurs quelle que soit leur version de logiciel grâce aux messages de changement de clé accompagnant le patch. Chaque décodeur dispose d'une clé publique dans la liste capable de décrypter une mise à jour de la version courante après neutralisation des anciennes clés.

Lors d'une mise à jour du logiciel d'un décodeur d'une version N à une version R où la différence R-N devient grande, par exemple supérieure à 10, il devient fastidieux pour un décodeur qui a une version R-1 de décrypter systématiquement tous les messages afin de vérifier les ordres de désactivation. Ce décodeur va appliquer sa clé courante (R-1) à chacun de ces messages pour constater qu'il ne peut pas interpréter son contenu.

Une première solution consiste à introduire en clair dans l'en-tête des messages des index correspondant aux numéros des différentes versions. Cet index sert uniquement à éviter la décryption des messages qui ont été encryptés par une clé différente de la clé courante. Cet index ne sélectionne pas le rang de la clé courante, seul la décryption réussie d'un message avec ladite clé courante provoque l'avance d'un rang dans la liste de clés.

Selon une deuxième solution, l'empreinte du patch de mise à jour est encryptée successivement par toutes les clés privées des mises à jour précédentes. Ce procédé oblige une utilisation de chaque clé publique de la liste, l'une après l'autre,

pour décrypter la signature. Dans ce cas d'encryption en chaîne, contrairement au précédent, toutes les clés publiques doivent rester disponibles dans la mémoire EEPROM du décodeur. Par exemple, pour une mise à jour de la version 1 vers la version N, l'empreinte du patch P est encryptée avec une clé privée de la version N. L'ensemble est ensuite encrypté avec la clé privée de la version N-1, puis avec la clé de la version N-2 et ainsi de suite jusqu'à la version 1. La décryption nécessite donc l'utilisation successive des clés publiques K1 à KN-1 correspondantes aux mises à jour de la version 1 à la version N. L'arrêt de ce mécanisme itératif se fait par la reconnaissance d'une marque appropriée dans le résultat de la décryption.

Si l'on souhaite protéger les données de mise à jour, une manière de procéder consiste à utiliser une clé de session SK générée aléatoirement par le centre de gestion par exemple. Pour des raisons opérationnelles de rapidité, cette clé est de type symétrique. Le centre de gestion encrypte le patch avec la clé de session SK et compose un ensemble de données comprenant la clé de session SK et l'empreinte du patch de mise à jour. Cet ensemble est encrypté par la clé privée courante de l'opérateur pour constituer le bloc de contrôle.

Le patch crypté et le bloc de contrôle sont chargés dans la mémoire vive RAM du décodeur. Le bloc est décrypté par la clé publique courante dans la liste ce qui donne l'empreinte du patch et la clé de session SK. Cette dernière est appliquée au patch chargé dans la mémoire RAM permettant sa décryption. Puis l'empreinte du patch est vérifiée et en cas de concordance le patch est installé dans la mémoire non volatile Flash.

La clé de session peut être introduite comme moyen de sécurisation supplémentaire dans l'une ou l'autre des variantes décrites plus haut par exemple:

- la mise à jour simple d'une version 1 à une version N en plusieurs étapes,
- la mise à jour d'une version N à R à l'aide d'un patch et des messages de désactivation des clés.

Dans un décodeur ayant été mis à jour à de nombreuses reprises, le nombre de clés publiques à disposition diminue tandis que le nombre de clés désactivées augmente en même temps que les mises à jour réussies. Afin de reconstituer une liste de clés pour permettre les futures mises à jour, une nouvelle liste de clés publiques peut être

envoyée au décodeur par le centre de gestion. Cette liste peut être incorporée dans le flux de données et accompagnée d'une signature comme pour un patch de mise à jour. Elle est stockée dans la mémoire EEPROM et remplace l'ancienne liste contenant des clés désactivées.

Selon une variante de la méthode de l'invention, le centre de gestion et les décodeurs disposent respectivement d'une liste de clés privées et publiques fixe. A chaque mise à jour, le centre de gestion choisit aléatoirement un ensemble de clés privées parmi celles de la liste et encrypte l'empreinte du patch successivement avec chaque clé de l'ensemble. Le centre compose un bloc de données comprenant l'empreinte encryptée (signature) et une suite de numéros correspondant aux rangs des clés choisies auparavant. Cette suite peut être transmise en clair ou encryptée avec une clé de session. Le décodeur recevant la suite de numéros sélectionne dans la liste des clés publiques, d'après leur rang, les clés nécessaires pour décrypter l'empreinte du patch. La liste ne peut pas comprendre plus d'une fois le même numéro de clés et la longueur de cette liste (nombre de clés utilisées) est connue et non modifiable. Dans cette variante, les listes de clés restent fixes et ne sont pas altérées suite à une installation réussie d'un patch. A chaque mise à jour, une nouvelle combinaison de clés prises dans la liste est utilisée pour la signature du patch. Un tiers devra donc toujours disposer d'un ensemble de clés pour introduire une mise à jour dans un appareil, ce qui nécessite des moyens plus conséquents que pour la détermination d'une seule clé.

Le procédé de sécurisation de mise à jour selon l'invention est indépendant du mode de transmission utilisé entre un fournisseur et un utilisateur. En effet, le procédé peut aussi s'appliquer sur des patchs distribués sur CD-ROM, sur disquette ou sur tout autre support de données numériques.

REVENDICATIONS

1. Méthode de sécurisation de mise à jour de données d'une pluralité d'appareils, chaque appareil recevant les mises à jour d'un centre de gestion, ces mises à jour comprenant des données appelées patch accompagnées d'un bloc de contrôle encrypté par une clé privée asymétrique prise dans une liste de clés contenue dans le centre de gestion, caractérisée par les étapes suivantes:

- sélection par l'appareil d'une clé courante dans une liste de clés publiques,
- réception du patch de mise à jour et stockage en mémoire,
- réception du bloc de contrôle encrypté,
- décryption dudit bloc par la clé publique courante,
- vérification que le bloc de contrôle décrypté corresponde audit patch,
- installation du patch reçu,
- désactivation de la clé courante et sélection de la clé suivante dans la liste.

2. Méthode selon la revendication 1 caractérisée en ce que le bloc de contrôle comprend une signature sur les données du patch, cette signature étant le résultat d'une fonction de hachage, et en ce que la vérification du bloc comprend l'étape d'établissement de la signature sur le patch reçu et la comparaison avec la signature décryptée dans le bloc de contrôle.

3. Méthode selon la revendication 1 ou 2, caractérisée en ce que le bloc de contrôle comprend une clé de session symétrique (SK) déterminée par le centre de gestion, cette clé étant utilisée pour encrypter les données du patch.

4. Méthode selon la revendication 1 ou 2, caractérisée en ce qu'à chaque mise à jour une nouvelle clé publique, issue de la liste, est utilisée par l'appareil.

5. Méthode selon les revendications 1 à 4, caractérisée en ce que la clé publique est détruite de la liste après son utilisation, ladite clé devenant inutilisable pour des mises à jour postérieures.

6. Méthode selon les revendications 1 à 5, caractérisée en ce que les clés publiques de la liste sont utilisées séquentiellement dans un ordre prédéterminé lors de chaque mise à jour.

7. Méthode selon les revendications 1 à 6, caractérisée en ce que la liste des clés publiques est stockée dans une mémoire non volatile, une clé utilisée pour une mise à jour est effacée définitivement de la mémoire autorisant l'accès à la clé suivante pour la prochaine mise à jour.

8. Méthode selon les revendications 1 à 7 caractérisée en ce que, pour la mise à jour du logiciel d'un appareil d'une version N à une version R, avec une différence R-N entre la nouvelle version et la précédente excédant l'unité, au moins un message Mn encrypté avec une clé privée PKn est ajouté permettant de changer la clé courante Kn à la clé suivante Kn+1 dans la liste, la décryption réussie dudit message provoquant la désactivation de la clé courante Kn et la sélection de la clé suivante Kn+1.

9. Méthode selon la revendication 8, caractérisée en ce que le nombre de messages M correspond au nombre de mise à jour séparant la version initiale de l'appareil et la version finale de la mise à jour.

10. Méthode selon les revendications 1 et 2, caractérisée en ce qu'une installation d'une mise à jour est suivie par une incrémentation d'un compteur ou un déplacement d'un pointeur indiquant le rang de la clé à sélectionner dans la liste lors de la prochaine mise à jour, la liste des clés étant inchangée.

11. Méthode selon les revendications 1 à 4, caractérisée en ce que le bloc de contrôle est encrypté successivement par les clés des mises à jour précédentes, chaque clé de la liste étant utilisée l'une après l'autre pour décrypter la signature.

12. Méthode selon les revendications 1 à 10, caractérisée en ce que les appareils consistent en des décodeurs de télévision à péage, la mise à jour d'un décodeur étant effectuée par le téléchargement, à partir d'un centre de gestion, d'un patch accompagné d'un bloc de contrôle, ledit bloc est stocké dans une mémoire vive (RAM), et est décrypté avec une clé publique courante d'une liste contenue dans une première mémoire non volatile du décodeur, puis vérifié et en cas de concordance, une commande provoque l'installation du patch dans une seconde mémoire non volatile et la désactivation de la clé courante.

13. Méthode selon la revendication 12, caractérisée en ce qu'une nouvelle liste de clés publiques est transmise au décodeur, ladite liste remplace la liste contenue dans la première mémoire contenant des clés désactivées par des mises à jour réussies précédemment.

ABRÉGÉ

La présente invention propose une méthode de sécurisation de mise à jour de logiciels d'une pluralité de décodeurs basée sur la génération d'une signature au moyen d'une clé privée asymétrique.

La mise à jour d'un décodeur étant effectuée par le téléchargement, à partir d'un centre de gestion, d'un bloc de données comprenant un patch et sa signature, ledit bloc est stocké dans une mémoire vive RAM. La signature est décryptée avec une clé publique courante d'une liste contenue dans une première mémoire non volatile du décodeur, puis vérifiée et en cas de concordance, une commande provoque l'installation du patch dans une seconde mémoire non volatile Flash et la désactivation de la clé courante.

Le but de la présente invention est de réduire considérablement l'impact de la découverte d'une clé privée par une analyse systématique du fonctionnement du logiciel du décodeur ou d'accroître notablement le temps et les moyens nécessaires au processus utilisé pour sa détermination.

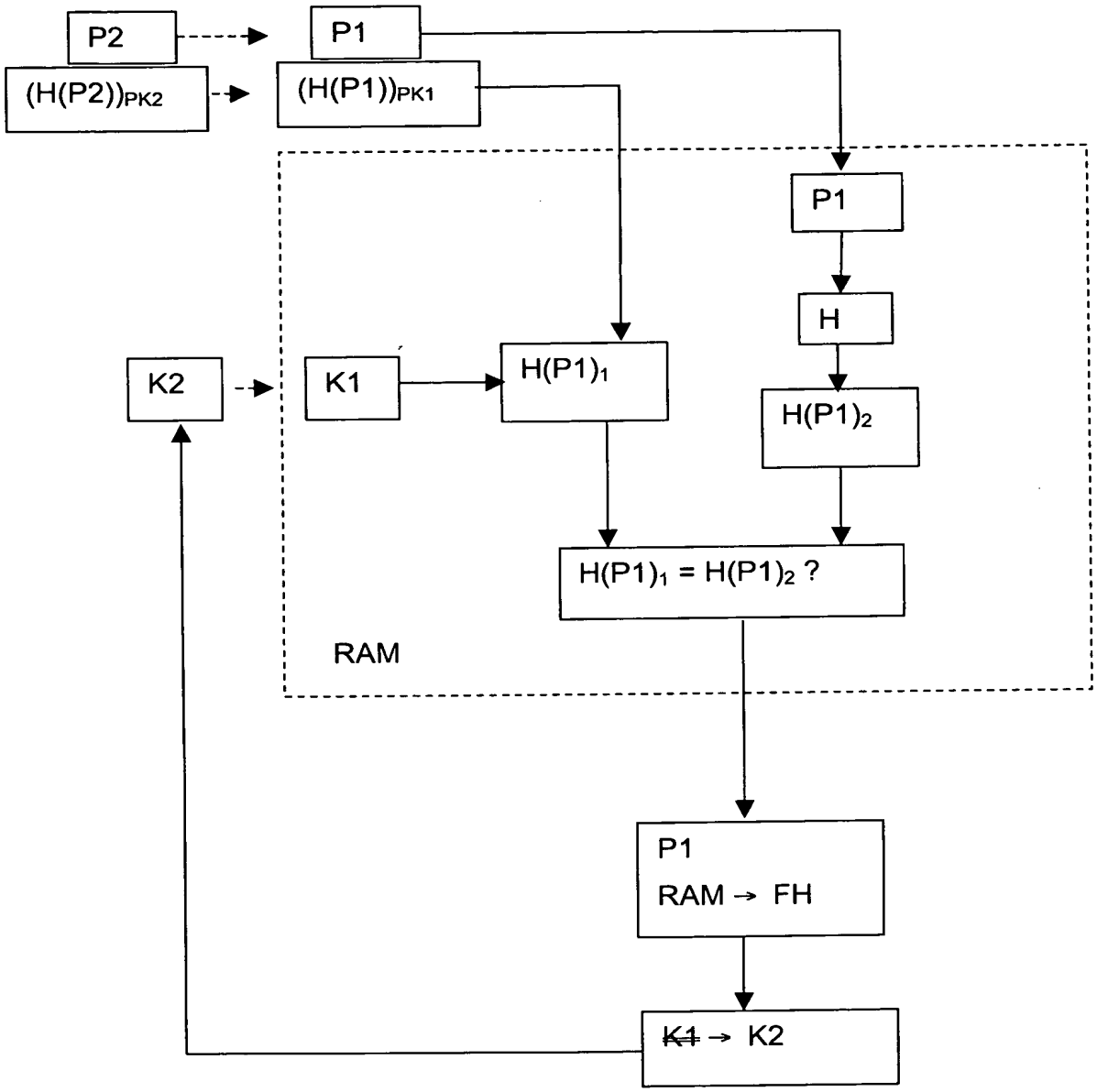

Fig. 1

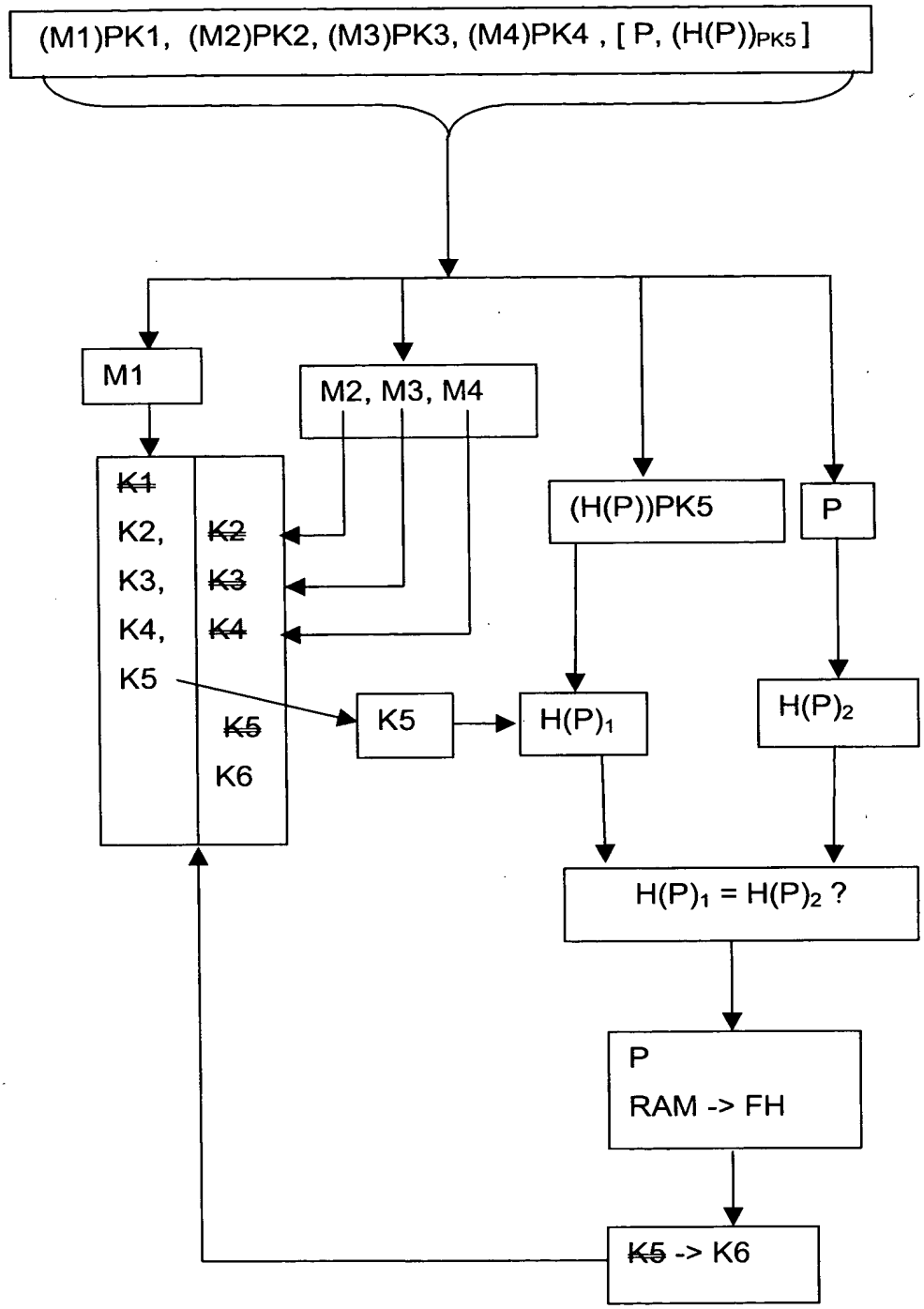

Fig. 2